

AML Summary

Lukas Diebold

December 15, 2025

Contents

1	Math Preliminaries	2
2	Conceptual Foundation	3
2.1	What is Machine Learning?	3
2.2	Conceptual foundation of inference	3
2.3	Artificial Intelligence	3
2.4	Extracting Value from Data - What is the problem?	4
2.5	What does Generalization mean?	4
2.6	Conceptional Foundation of Inference	4
3	Fundamentals of Machine Learing	5
3.1	Efficiency: Rao Cramer bound	5
4	Regression	7
4.1	Act 1: High-dimensional regression is unstable	7
4.2	Act 2: Stability via Regularization	7
4.3	Act 3: Polynomial regression via kernels	8
4.4	Act 4: Neural Networks	8

1 Math Preliminaries

Gibbs Distribution The Gibbs distribution is a probability distribution that assigns likelihoods to states based on a cost function, with lower-cost states being more probable. Given a set of states $x \in \mathcal{X}$, a cost function $E(x)$, and an inverse temperature parameter $\beta > 0$, the Gibbs distribution is

$$p(x) = \frac{1}{Z} e^{-\beta E(x)}$$

where the partition function Z ensures normalization

$$Z = \sum_{x \in X} e^{-\beta E(x)}$$

2 Conceptual Foundation

2.1 What is Machine Learning?

"ML is a mathematization of epistemology!". In philosophy, it is the science of knowledge, the science of what can be known. This is relevant, because in ML we are interested in systems that produce/generate knowledge.

The goal is then to observe 'reality' and draw conclusions from the observations. This can be seen as a perception-action cycle. Where perception is the result of our observations (typically in a data space \mathcal{X}) and the actions are part of a hypothesis space. To go from the data space to the hypothesis class \mathcal{C} we generally use an algorithm A . See Figure 1 for an overview.

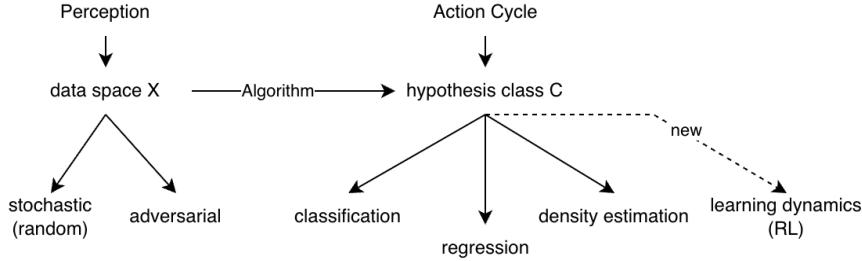


Figure 1: Overview ML

Information processing occurs when $|\mathcal{X}| \gg |\mathcal{C}|$. Taking the example of combinatorial optimization problems, $|\mathcal{X}|$ would be the space of weighted graphs, and we look for a color of the graph or something similar. Then $|\mathcal{X}| \approx K^{\binom{n}{2}}$ and $|\mathcal{C}| \approx e^{n \log n}$ so we observe that $|\mathcal{X}|$ is much larger.

2.2 Conceptual foundation of inference

1. Perception of reality is mediated by data of senses/sensors
2. Data are stochastic \rightarrow probabilistic
3. Sensing restricts us to selected aspects of reality
4. Humans interpret data by a huge reduction in degrees of freedom
 $|x| \gg |e|$ (space of graphs \gg space of colorings, cycles, spanning trees)
5. Tuple ($\{\text{data}\}$, $\{\text{hypotheses}\}$) define models:

$$\begin{aligned} \mathcal{A} : \mathcal{X} &\longrightarrow \mathcal{C} \\ x &\longmapsto c = \mathcal{A}(x) \end{aligned}$$

6. Experiments ϵ provide us with data
7. Learning means interpreting X w.r.t hypotheses \mathcal{C}

2.3 Artificial Intelligence

Taking a high level view. We live in very high dimensional data, which we are not able to fully process. We use algorithms to make sense of the data, and this then informs our values, from this we get the following relation

$$\text{Data} \longrightarrow \text{Algorithms } \mathcal{A} \longrightarrow \text{Values}$$

In epistemology, we differentiate between **deduction** and **induction**. Deduction is a form of reasoning in which the conclusion follows necessarily from the premises, while induction tries to generalize, that is, the conclusion goes beyond the premises and generally probabilistic. We can construct a model in which deduction and induction form a **feedback loop**, not two isolated

methods. In simpler terms, on one side we try to formulate axioms from our observations (empirical data), and on the other side we then use these axioms and derive logical consequences from them. This is not anything new, and this cycle generally informs the model of "Theory, Experiment, Computation" in science. What has changed with ML/AI is that we're now in the era of non-parametric modeling.

2.4 Extracting Value from Data - What is the problem?

- Algorithms that process inputs with noise compute random variables as outputs!
- Algorithms should compute typical solutions!
- When do algorithms generalize over noise/model mismatch?
- How can algorithms autonomously improve performance?

2.5 What does Generalization mean?

- Out-of-sample risk

$$\theta^*(X') \sim \mathbb{P}^A(\theta | X) \in \arg \min_{\mathbb{P}(\cdot | \cdot)} \mathbb{E}_{X'} \mathbb{E}_{\theta | X'} \mathbb{E}_{X'' | X'} R(\theta, X'')$$

where X' is the training data and X'' is the test data, so this is the risk where θ is conditioned on the training data (trained the model).

- Log loss of posterior (risks and probabilities are dependent!)

$$\begin{aligned} \theta^*(X') \sim \mathbb{P}^A(\theta | X) &\in \arg \min_{\mathbb{P}(\cdot | \cdot)} \mathbb{E}_{X'} \mathbb{E}_{\theta | X'} \mathbb{E}_{X'' | X'} (-\log \mathbb{P}(\theta | X'')) \\ &\in \arg \min_{\mathbb{P}(\cdot | \cdot)} \mathbb{E}_{X'} \mathbb{E}_{\theta | X'} \mathbb{E}_{X'' | X'} (\beta R(\theta, X'') + \log Z) \end{aligned}$$

- Posterior agreement

$$\theta^*(X') \sim \mathbb{P}^A(\theta | X) \in \arg \min_{\mathbb{P}(\cdot | \cdot)} \mathbb{E}_{X'} \mathbb{E}_{X'' | X'} (-\log \mathbb{E}_{\theta | X'} \mathbb{P}(\theta | X''))$$

2.6 Conceptional Foundation of Inference

- Our perception of "our world" / reality is mediated by data of senses / sensors.
- Our data are influenced by chance.
- Creatures interpret selected aspects of reality by hypotheses to "survive and reproduce"
- Data and hypotheses define models to enable judgements, decisions and actions.
- AI / ML: Algorithms define relations of data and hypotheses, e.g., they select models !

3 Fundamentals of Machine Learning

Bayes Rule:

$$\mathbb{P}(\text{model} \mid \text{data}) = \frac{\mathbb{P}(\text{ data} \mid \text{ model })\mathbb{P}(\text{ model })}{\mathbb{P}(\text{ data })}$$

ML method:

$$\widehat{\text{model}}_m \in \arg \max_{\text{model}} \mathbb{P}(\text{data} \mid \text{model})$$

where $\widehat{\text{model}}_n$ is consistent, asymptotically normal, and asymptotically efficient

consistency: a point estimated $\hat{\theta}_n$ of the parameter $\theta = \theta_0$ is consistent if

$$\forall \varepsilon > 0, \mathbb{P}\left(\left|\hat{\theta}_n - \theta_0\right| > \varepsilon\right) \xrightarrow{n \rightarrow \infty} 0$$

more formally

$$\forall \theta, \forall \varepsilon, \delta > 0, \exists n_0, \forall n > n_0, \mathbb{P}\left(\left|\hat{\theta}_n - \theta\right| < \varepsilon\right) > 1 - \delta$$

efficiency:

$$\hat{\theta}_n = \arg \min_{\hat{\theta}} \mathbb{E}\left[\left(\hat{\theta}_n - \theta_0\right)^2\right]$$

3.1 Efficiency: Rao Cramer bound

Problem: Precision of parameters estimation.

Given the likelihood $p(y \mid \theta)$ for $\theta \in \Theta$, data $y_1, \dots, y_n \sim p(y \mid \theta = \theta_0)$ we are interested in the question: "How precisely can we estimate $\theta = \theta_0$ given n samples?". To answer this we define an estimator $\hat{\theta}(y_1, \dots, y_n)$ and estimate the expected deviation

$$\mathbb{E}_{y \mid \theta}[(\hat{\theta} - \theta)^2]$$

Score: $\Lambda = \frac{\partial}{\partial \theta} \log p(y \mid \theta) = \frac{\frac{\partial}{\partial \theta} p(y \mid \theta)}{p(y \mid \theta)}$

bias : $b_{\hat{\theta}} = \mathbb{E}_{y \mid \theta}[\hat{\theta}(y_1, \dots, y_n)] - \theta$

Expected score :

$$\begin{aligned} \mathbb{E}_{y \mid \theta}[\Lambda] &= \int p(y \mid \theta) \frac{\frac{\partial}{\partial \theta} p(y \mid \theta)}{p(y \mid \theta)} dy \\ &= \frac{\partial}{\partial \theta} \int p(y \mid \theta) dy = \frac{\partial}{\partial \theta} 1 = 0 \end{aligned}$$

Expected score times estimator product :

$$\begin{aligned} \mathbb{E}_{y \mid \theta}[\Lambda \hat{\theta}] &= \int p(y \mid \theta) \frac{\frac{\partial}{\partial \theta} p(y \mid \theta)}{p(y \mid \theta)} \hat{\theta} dy \\ &= \frac{\partial}{\partial \theta} \left(\int p(y \mid \theta) \hat{\theta} dy - \theta \right) + 1 \\ &= \frac{\partial}{\partial \theta} (\mathbb{E}_{y \mid \theta} \hat{\theta} - \theta) + 1 = \frac{\partial}{\partial \theta} b_{\hat{\theta}} + 1 \end{aligned}$$

Cross correlation between score and estimator :

$$\mathbb{E}_{y \mid \theta}[(\Lambda - \underbrace{\mathbb{E}\Lambda}_{=0})(\hat{\theta} - \mathbb{E}\hat{\theta})] = \mathbb{E}_{y \mid \theta}[\Lambda \hat{\theta}] - \underbrace{\mathbb{E}_{y \mid \theta}[\Lambda] \mathbb{E}\hat{\theta}}_{=0}$$

Cauchy-Schwarz inequality :

$$\begin{aligned}
& \left(\mathbb{E}_{y|\theta} [\Lambda(\hat{\theta} - \mathbb{E}\hat{\theta})] \right)^2 \leq \mathbb{E}_{y|\theta} [\Lambda^2] \mathbb{E}_{y|\theta} [(\hat{\theta} - \theta - \mathbb{E}\hat{\theta} + \theta)^2] \\
& = \mathbb{E}_{y|\theta} [\Lambda^2] \left(\mathbb{E}_{y|\theta} [(\hat{\theta} - \theta)^2] - \mathcal{L}(\mathbb{E}\hat{\theta} - \theta)^2 + (\mathbb{E}\hat{\theta} - \theta)^2 \right) \\
& = \mathbb{E}_{y|\theta} [\Lambda^2] \left(\mathbb{E}_{y|\theta} [(\hat{\theta} - \theta)^2] - b_{\hat{\theta}}^2 \right) \\
& \mathbb{E}_{y|\theta} [(\hat{\theta} - \theta)^2] \geq \frac{\left(\mathbb{E}_{y|\theta} [\Lambda\hat{\theta}] \right)^2}{\mathbb{E}_{y|\theta} [\Lambda^2]} + b_{\hat{\theta}}^2 = \frac{\left(\frac{\partial}{\partial\theta} b_{\hat{\theta}} + 1 \right)^2}{\mathbb{E}_{y|\theta} [\Lambda^2]} + b_{\hat{\theta}}^2
\end{aligned}$$

- $\hat{\iota}$ General Rao Cramer bound for estimator $\hat{\theta}$

$$\mathbb{E}_{y|\theta} [(\hat{\theta} - \theta)^2] \geq \frac{\left(\frac{\partial}{\partial\theta} b_{\hat{\theta}} + 1 \right)^2}{\mathbb{E}_{y|\theta} [\Lambda^2]} + b_{\hat{\theta}}^2$$

Fisher information : $\mathbb{E}_{y|\theta} [\Lambda^2] = \int \rho(y | \theta) \left(\frac{\partial}{\partial\theta} \log \rho(y | \theta) \right)^2 dy =: I(\theta)$

Remarques

II) Note tradeoff $\frac{\partial}{\partial\theta} b_{\hat{\theta}} < 0$ vs. $b_{\hat{\theta}}^2$ for biased estimators! unbiased estimators ought not be the best estimators!

case with n samples

$$\begin{aligned}
\mathbb{E}_{y_1, \dots, y_n | \theta} [\Lambda^2] &= \int p(y_1, \dots, y_n | \theta) \underbrace{\frac{\partial}{\partial\theta} \log p(y_1, \dots, y_n | \theta)}_{= \frac{\partial}{\partial\theta} \sum_{i=n} \log p(y_i | \theta) = \sum_{i=n} \Lambda_i})^2 dy_1 \dots dy_n =: I^{(n)}(\theta) \\
&= \int p(y_1, \dots, y_n | \theta) \underbrace{\left(\sum_{i=n} \Lambda_i^2 + \sum_{i=n} \sum_{j=n} \Lambda_i \Lambda_j \right)}_{=0} dy_1 \dots dy_n \\
&= \sum_{i=n} \int p(y_i | \theta) \Lambda_i^2 dy_i = nI(\theta)
\end{aligned}$$

Remark: The Fisher information of n iid. r.v. is $n \times$ Fishes information of 1 r.v.

4 Regression

4.1 Act 1: High-dimensional regression is unstable

we assume X and y are distributed according to a distribution p_* (i.e. $X, y \sim p_*$) and

$$y = f_*(x) + \varepsilon \quad \text{with } \varepsilon \sim \mathcal{N}(0, \sigma)$$

Then our task is to estimate f_* from $D = \{x_i, y_i\} \sim p_*$. The problem in this form is not tractable, this is why we restrict the choice of functions. First we limit ourselves to linear functions, that is functions of the form

$$f_*(x) = \beta^\top x$$

This problem we then solve using Maximum Likelihood Estimation (MLE), that is

$$\begin{aligned} \hat{\beta} &= \arg \max_{\beta \in \mathbb{R}^n} p(D | \beta) \\ &= \dots \\ &= \arg \min \text{MSE}(D, \beta) \\ &= \frac{1}{n} \sum_{i \leq n} (y_i - \beta^\top x_i)^2 \\ &= (X^\top X)^{-1} X^\top y \\ &= \dots \\ &= X^\top (X X^\top)^{-1} y \end{aligned}$$

which is known as the ordinary least squares estimator, where

$$X = \begin{bmatrix} -x_1 - \\ \vdots \\ -x_n - \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

This estimator has some interesting properties. It is unbiased and, by the Gauss-Markov Theorem, it is the best linear unbiased estimator (BLUE), i.e. it attains the smallest variance among all linear unbiased estimators. Thus, from the formula

$$\text{error} = \text{bias}^2 + \text{variance} + \text{noise}$$

we find that this estimator is the one with the smallest error of all the unbiased estimators. Then why does no-one use this estimator? If we introduce a bit of bias, we can significantly reduce the variance. So what is $\text{Var}(\hat{\beta})$? For this we use the SVD of $X = UDV^\top$; if we plug this into the formula from above we get

$$\hat{\beta} = V D^{-1} U^\top y$$

so, since y has a gaussian distribution and we just multiply it with a matrix $(VD^{-1}U^\top)$, $\hat{\beta}$ also has a gaussian distribution. Then

$$\text{Var}(\hat{\beta}) = \dots = \sigma^2 V D^{-2} V^\top = \sigma^2 \sum_{i \leq n} \frac{1}{D_{ii}^2} V_i V_i^\top$$

What are the implications of this? In high-dimensional data many features are correlated, so X is close to low rank and several singular values D_{ii} are tiny. The $1/D_{ii}^2$ factors then explode, making the variance (and with it prediction instability) very large, even though the estimator remains unbiased.

4.2 Act 2: Stability via Regularization

The typical process of **Bayesian inference** goes through the following stages:

1. Prior $\beta \sim \mathcal{N}(0, \tau^2 I)$
2. Observe likelihood $p(D|\beta)$

$$3. \text{ Posterior (adjustment of the likelihood)} p(\beta|D) \propto p(\beta)p(D|\beta) \propto \exp\left(-\frac{1}{2\sigma^2} \text{MSE}(D, \beta) - \frac{1}{2\tau^2} \|\beta\|^2\right)$$

Observe that the last term is equivalent to the loss induced by **ridge regression**. With a Laplace prior (heavy tails) on β we obtain **lasso** instead. The prior variance τ^2 controls the amount of shrinkage and therefore the bias-variance trade-off. If we do the math we get

$$\hat{\beta}_{\text{MAP}} = \left(x^\top x + \frac{\sigma^2}{\tau^2} I \right)^{-1} x^\top y$$

and if we do again an SVD to calculate the variance we get

$$\text{Var}(\hat{\beta}_{\text{MAP}}) = \sigma^2 \sum_{i \leq n} \frac{D_{ii}^2}{\left(D_i^2 + \frac{\sigma^2}{\tau^2}\right)} V_i V_i^\top$$

Compared to OLS, the shrinkage factor $\frac{D_{ii}^2}{D_i^2 + \sigma^2/\tau^2}$ down-weights directions with small singular values, reducing variance at the cost of introducing bias.

4.3 Act 3: Polynomial regression via kernels

Now we change our assumption for $f_*(x)$ so that it is a polynomial function on x . We have

$$f_*(x) = \varphi(x)^\top \beta_*$$

where $\beta_* \in \mathbb{R}^\infty$ and

$$\varphi(X) = K_x \left(\frac{x_1^{\alpha_1} \dots x_d^{\alpha_d}}{\sqrt{\alpha_1! \dots \alpha_d!}} \right)_{\alpha \in \mathbb{N}^d}$$

Then for $x, x' \in \mathbb{R}^a$

$$\begin{aligned} \varphi(x)^\top \varphi(x') &= K_{RBF}(x, x') \\ &= \exp\left(-\frac{1}{2} \|x - x'\|^2\right) \end{aligned}$$

and

$$\begin{aligned} \hat{\beta} &= \arg \min_{\beta \in \mathbb{R}^\infty} \frac{1}{n} \sum_{i \leq n} (y_i - \varphi(x_i)^\top \beta)^2 \\ &= \Phi^\top (\Phi \Phi^\top)^{-1} y \end{aligned}$$

where

$$\Phi = \begin{bmatrix} \varphi(x_1)^\top \\ \varphi(x_2)^\top \\ \vdots \\ \varphi(x_n)^\top \end{bmatrix} \in \mathbb{R}^{n \times \infty}$$

This doesn't look good. But let x_* be a test point. Then

$$\begin{aligned} \hat{y}_* &= \varphi(x_*)^\top \hat{\beta} \\ &= \varphi(x_*)^\top \Phi^\top (\Phi \Phi^\top)^{-1} y \\ &= k(x_*)^\top K^{-1} y \end{aligned}$$

where $k(x_*) = (\varphi(x_*)^\top \varphi(x_i))_{1 \leq i \leq n}$ and $K_{ij} = (\varphi(x_i)^\top \varphi(x_j))$. This is the kernel trick: we only need inner products via K , not the infinite-dimensional $\varphi(\cdot)$.

The problem is that the inversion of the matrix is $O(n^3)$, which becomes costly for large datasets even though we avoided the infinite feature map explicitly.

4.4 Act 4: Neural Networks

We assume f_* has only a single, very wide hidden layer.

$$f_*(X) = \frac{1}{\sqrt{m}} \sum_{i \leq m} \alpha_i \phi(\omega_i^\top X)$$

where ϕ is a nonlinear activation function (e.g. ReLU, tanh), and the network has m hidden units. The parameters are $\theta = \{\alpha_i, w_i\}_{i \leq m}$, i.e. both the output weights α_i and the input weights w_i are learned. We initialize with

$$\theta_0 \sim \mathcal{N}(0, w^2)$$

and we update our parameters using gradient descent.

$$\theta_{t+1} \leftarrow \theta_t - \eta \nabla_\theta \text{MSE}(D, \theta_t)$$

The gradient can be written in matrix form as

$$\nabla_\theta \text{MSE}(D, \theta_t) = \tilde{\Phi}_t^\top (f_t - y)$$

where

$$\tilde{\Phi}_t = \left(-\nabla_\theta f(x_i; \theta_t)^\top \right)_{i \leq n} \in \mathbb{R}^{n \times |\theta|} \quad \text{and} \quad f_t = (f(x_i; \theta_t))_{i \leq n} \in \mathbb{R}^n$$

Here $\tilde{\Phi}_t$ is the feature matrix whose i -th row is the gradient of the network output with respect to all parameters, evaluated at data point x_i and current parameters θ_t .

In the *lazy training regime* (small learning rate, wide network), the parameters stay close to initialization, so we can linearize the network via a first-order Taylor expansion around θ_0 :

$$f_t \approx f_0 + \tilde{\Phi}_0 (\theta_t - \theta_0)$$

Assuming the feature matrix $\tilde{\Phi}_t$ remains approximately constant at $\tilde{\Phi}_0$ (which holds when $m \rightarrow \infty$), gradient flow yields

$$\theta_t - \theta_0 = \tilde{\Phi}_0^\top \left(\tilde{\Phi}_0 \tilde{\Phi}_0^\top \right)^{-1} (f_t - f_0)$$

This says the parameter change lies in the span of the gradients and is chosen to optimally fit the training residuals.

Now let x_* be a test point. Plugging the linearization into the prediction yields

$$f_t(x_*) \approx f_0(x_*) + \nabla_\theta f(x_*, \theta_0)^\top \tilde{\Phi}_0^\top \left(\tilde{\Phi}_0 \tilde{\Phi}_0^\top \right)^{-1} (f_t - f_0)$$

Define the *neural tangent kernel (NTK)* K with entries

$$K_{ij} = \nabla_\theta f(x_i, \theta_0)^\top \nabla_\theta f(x_j, \theta_0) = \left[\tilde{\Phi}_0 \tilde{\Phi}_0^\top \right]_{ij}$$

and similarly $k(x_*) = (\nabla_\theta f(x_*, \theta_0)^\top \nabla_\theta f(x_i, \theta_0))_{i \leq n}$.

In the infinite-width limit ($m \rightarrow \infty$), the random initialization ensures $f_0(x) \rightarrow 0$ for all x (the outputs average out), and after infinite training time ($t \rightarrow \infty$), gradient descent drives the training residual to zero so $f_t \rightarrow y$. The prediction becomes

$$f_\infty(x_*) = k(x_*)^\top K^{-1} y$$

This is exactly the result we obtained in Act 3 for kernel regression.

Conclusion: Gradient descent on a very wide neural network operates in a *kernel regime*, where training is equivalent to kernel ridge regression with the neural tangent kernel. The NTK is determined by the architecture and activation function, but the solution has the same closed-form structure $k(x_*)^\top K^{-1} y$ as any other kernel method. In practice, finite-width networks can escape this regime and learn richer, feature-learning representations—this lazy limit is a useful theoretical baseline.